제18회 임베디드SW경진대회 개발완료보고서

[webOS 기반 차량용 인포테인먼트 솔루션]

□ 개발 요약

팀 명	하이멕

차량 옆면 구현

AR, 객체 인식을 이용한
효과적인 인포테인먼트 제공

작품명	커넥티드 윈도우
작품설명 (요약)	차량 주변의 정보와 AR컨텐츠를 통해 차별된 인포테인먼트를 측면 창문 디스플레이로 사용자에게 제공한다.
소스코드	https://github.com/HyunCello/2020EBSWContest_webOS_4002
시연동영상	https://youtu.be/PABgotuMUj4

□ 개발 개요

○ 개발 작품 동기

현재 차량에서 정보 제공은 운전자 중심이기 때문에 나머지 탑승자들은 차와 관련된 정보를 얻을 기회가 부족하다. 하지만 발전하는 차량의 형태를 생각해볼 때 점점 사람이 운전하지 않는 환경이 더욱 더 많이 조성될 것이기 때문에 운전자 중심의 정보제공을 넘어서는 솔루션이 필요하게 되었다. 이를 위해 차량 및 주변과 관련된 정보를 운전자뿐만 아니라 탑승객 모두가 자동차 내부와 외부에 관련된 정보를 얻을 수 있는 시스템 구현을 목표로 하였다. 또한 엔터테인적 요소를 추가하여 사용자가 정보를 더욱 재미있게 받아들일 수 있는 것을 목표로 하였다.

현 시점에서 차량의 측면 창문은 주변을 볼 수 있는 단순한 기능만을 가지고 있다. 우리는 측면 창문을 단순한 기능을 넘어서서 주변의 정보, 장치들과 연결되어 정보를 주고받을 수 있고, 또한 엔터테인적인 요소를 더하여 유용한 방법으로 사용할 수 있게 하고자 하였다.

최근 LG 디스플레이에서 개발이 진행되고 있는 터치가 가능한 투명 디스플레이와 엔터테인적 요소가 강한 AR을 접목히게 된다면 차량 창문을 활용하여 인포테인먼트를 체감할 수 있는 '커넥티드 윈도우'라는 아이디어를 떠올리게 되었다. 투명 디스플레이는 현재 상태로는 사용하기 힘들어 터치디스플레이를 투명 디스플레이로 가정하고 커넥티드 윈도우의 기능을 구현하고자 하였다.

'커넥티드 윈도우'에는 차량 내-외부 정보를 손쉽게 얻을 수 있기에 여행을 떠나 활용할 때 진가를 발휘할 수 있다고 판단했다. 그래서 '제주도 여행'을 예시로 솔루션을 개발하여 타당성을 입증하고자 하였다.

○ 개발 작품 개요

‘커넥티드 윈도우’는 차량과 주변의 정보를 다른 방법을 거치지 않고 차 창문을 이용해서 손쉽게 바로 접근할 수 있도록 해주는 제품이다. 평소에 주변에 모르는 곳이나 교통상황이 궁금할 때는 휴대폰의 지도 어플을 사용해서 알아보고, 여행 테마를 탐색할 때는 여러 블로그들을 찾아보고, 트렁크 속 짐은 정차한 후 트렁크를 열어보는 등 궁금한 것에 대해서 여러 과정을 거쳐야 하지만 ‘커넥티드 윈도우’를 이용해서 단 한 번의 터치로 이를 해결할 수 있다.

○ 개발 목표

탑승자를 위한 기능은 총 6가지이다. 사용자 인식, 트렁크 확인, 목적지 설정, 교통상황 확인, 여행 테마 찾기, 주변 안내를 구현하고자 했다. 우선 개인 맞춤형 인포테인먼트를 제공하기 위해 차량 내부에 설치된 카메라를 이용하여 사용자의 얼굴을 인식하여 커넥티드 윈도우를 구동하고 이후 사용자의 눈꺼풀의 상태를 주기적으로 인식하여 사용자가 잠들거나 차에서 내렸으면 커넥티드 윈도우에서 실행되고 있는 창을 내릴 수 있도록 했다.

트렁크 확인 메뉴를 선택하면 트렁크의 카메라를 통해 Jetson TX2에서 짐을 인식하고 트렁크의 내용물이 캡처 되어 커넥티드 윈도우에 나타나게 하여 단 한 번의 터치로 짐을 확인할 수 있도록 했다.

목적지 설정을 선택하면 차 창문에 나타난 지도에서 원하는 목적지를 검색하여 설정할 수 있고, 교통상황 확인기능을 통해 지도를 통해 주변의 교통상황을 파악할 수 있다.

여행 테마 찾기 기능은 커넥티드 윈도우 상에서 제주도 관광 사이트로 연결하여 여행 테마별로 관광지를 찾아볼 수 있다. 마지막으로 주변안내 기능은 POI를 AR로 구현해 다량의 정보를 보다 흥미롭게 전달하는 것으로 ‘주변 안내’ 버튼을 선택하면 트렁크 짐들을 확인하는 방법과 비슷한 방식으로 Jetson TX2를 통해 주변 풍경을 인식하고 이 데이터를 ‘커넥티드 윈도우’에 불러와 AR을 이용해 컨텐츠에 대한 내용을 보다 친근하고 흥미롭게 표현하였다.

□ 개발 환경 설명

○ Hardware 구성

<완성 이미지>

- 뼈대 - 아크릴

제작 과정에서 3D 프린팅으로 전체를 출력하기엔 문제점들이 있었다. 가로 길이가 2m 에 가까워서 출력 시간이 너무 오래 걸린다는 점, 3D 프린터가 출력할 수 있는 크기로 조각 내어 출력 후 이어 붙이는데 피스들의 품질이 균일하지 않아서 틀어짐이 생긴다는 점이 가장 큰 문제였다. 그래서 아크릴로 뼈대를 만들고 나머지는 3D 프린터로 뼈대에 붙이는 방식을 선정했다.

뼈대는 모니터 등등의 장비들을 모두 거치해도 버틸 만큼 튼튼해야 했기 때문에 아크릴로 선택을 하였다. 뼈대의 크기가 830mm x 415mm 로 꽤 컸기 때문에 학교에 구비된 레이저 커팅기 사이즈를 초과해서 두개로 나누어 이어 붙여서 제작하기로 했다.

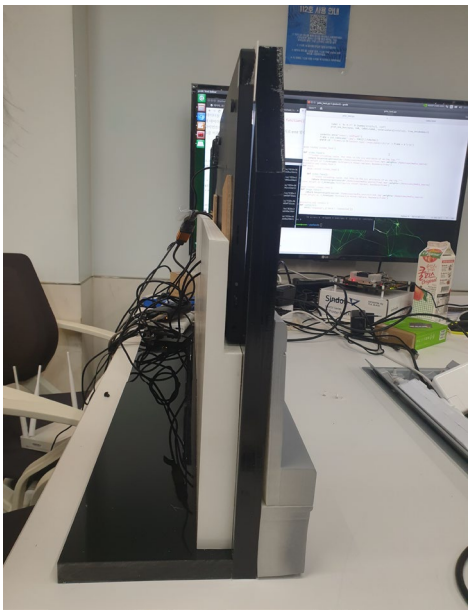
차량을 표현했다는 것을 사람들이 느낄 수 있게 하는 게 목표였고, 목표를 달성하는 한에서 최대한 압축적으로 만들고자 했다. 그래서 아크릴은 유광 흑색 아크릴을 사용했고 중심이 되는 창문(모니터)의 가장자리에 창틀의 형태(차 문의 아웃라인)로 재단했다.

- 모니터

Δ 뒷좌석 창문

Δ 앞 좌석 창문

15.6" 터치 디스플레이 두 대를 나란히 두어 차량의 단면을 보는 것처럼 앞 좌석 뒷좌석의 창문을 표현했다. 커넥티드 윈도우는 차 측면에 위치해서 터치를 통한 조작이 간편하기 때문에 터치 디스플레이를 선택했다.

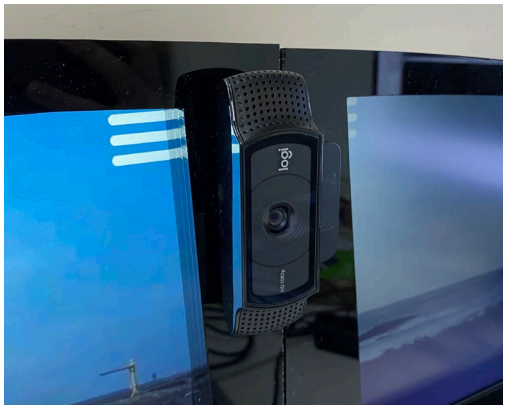


모니터를 뼈대에 거치하기 위해 나무 틀(흰색)을 뒤에 붙였다. 모니터를 간편하게 넣었다 뺐다 할 수 있고, 모니터 측면에 있는 버튼, 볼륨 버튼, 케이블 연결부가 개방되어 있어야 하며, 모니터가 고정되어야 하기 때문에 위와 같은 형태로 만들었다.

- 차 내부 표현재

차량 내부에서 보는 시점이라는 것을 전달하기 위해, 실제 차량 내부를 참고해 3D 프린터로 모형을 출력하였다. '차 내부'라는 것을 전달하기 위해 복잡하고 곡선이 많은 형태로 만들어야 해서 3D 프린팅이 가장 적합했다. 출력 시 변형을 줄이기 위해 abs 필라멘트를 사용했다

- 카메라

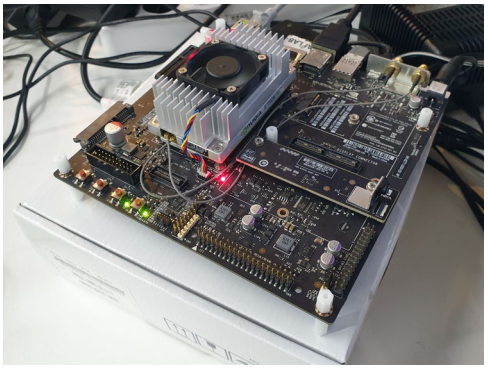


Logitech C920r 웹 캠을 사용했다. 이 카메라의 설치 목적은 사용자를 식별하고 사용자의 상태를 감지하기 위함이다. 구현 상황에선 뒷좌석에 있는 사람을 대상으로 설정했기 때문에 가운데에 위치시키고 뒷좌석 방향으로 각도를 틀어 고정했다. 또한 차량 형태를 최대한 해치지 않는 위치로 선정했다.

- Raspberrypi4

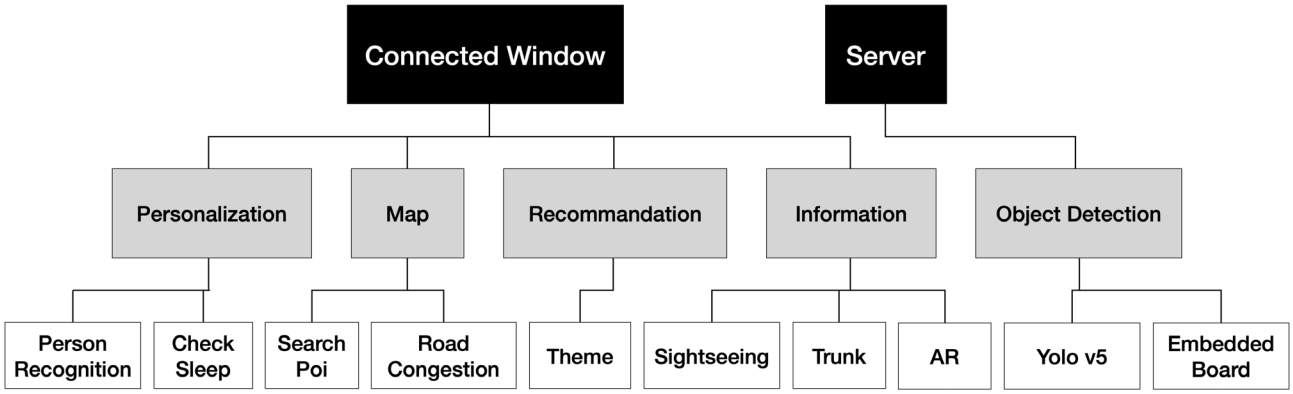
'커넥티드 윈도우'에서 컴퓨터의 본체의 역할을 하며 WebOS를 구동하며 Jetson TX2가 담당하는 영상처리 이외의 모든 작동을 담당한다.

- Nvidia Jetson TX2



딥러닝 객체인식과 영상처리를 위한 임베디드 보드이다. 개발 초기에는 Nvidia Jetson Nano 보드를 이용하여 개발하였지만 객체인식 연산속도를 높이기 위하여 TX2 보드로 변경하였다.

○ Software 구성



- Connected Window

차량 창문 표현

- Personalization

사용자를 인식하여 그에 맞는 컨텐츠 및 정보를 제공

⇒ Person Recogniton

⇒ Sleeping Check

- Map

지도 기능 중 일부를 제공하여 네비게이션의 기반 제공

⇒ Search POI

⇒ Check Road Congestion

- Recommandation

사용자가 필요한 정보 추천

⇒ Find Travel Theme

- Information

인지를 통해 정보 제공

⇒ Sightseeing + AR

⇒ Trunk Information

- Server

Connected Window에 보내줄 데이터 생성

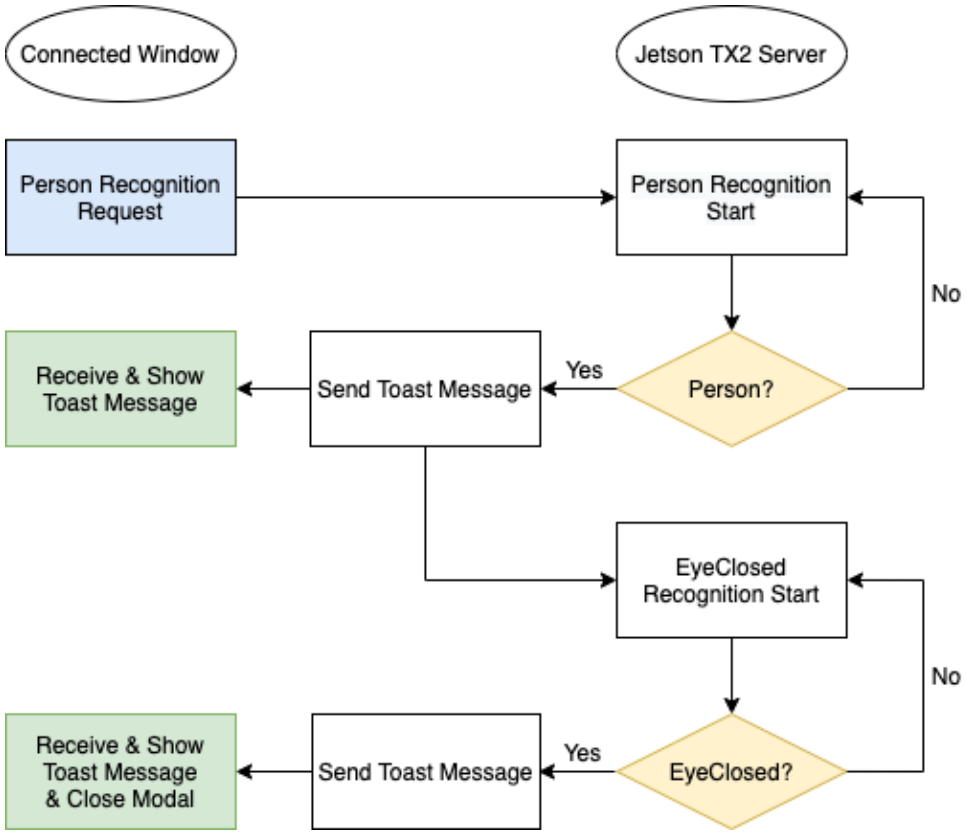
- Object Detection

ML을 이용하는 Object Detection 구현

⇒ YOLOv5 + Jetson TX2

○ Software 설계도

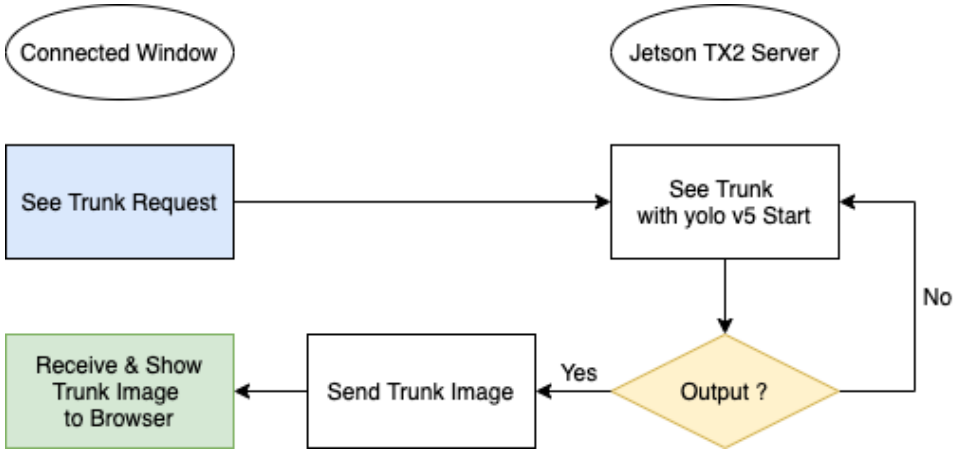
- 사용자 인식 및 수면여부 확인



- 컨텐츠 제공 전, 사용자의 얼굴을 식별하여 사용자 로그인 후 맞춤 컨텐츠 제공

- 컨텐츠 재생중에 주기적으로 사용자의 눈꺼풀을 확인하여 사용자가 수면에 들었을 시 컨텐츠를 종료

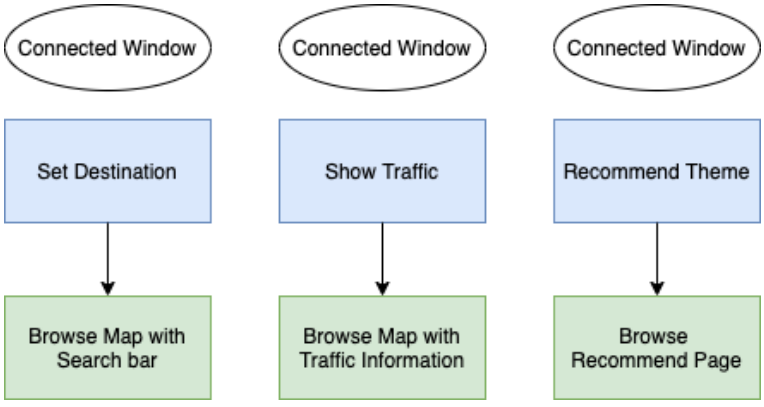
- 트렁크 확인



- 트렁크 확인

사용자가 서버(Jetson TX2)에 트렁크 확인을 요청(Request)하면 Jetson TX2에서 트렁크를 촬영한 이미지를 객체인식 알고리즘(YOLOv5)에 대입하여 연산한 후 이미지에 포함된 객체 정보와 객체의 위치를 Boundary Box로 표현하여 응답(Response)한다.

- 목적지 설정, 교통상황 확인, 추천테마 소개

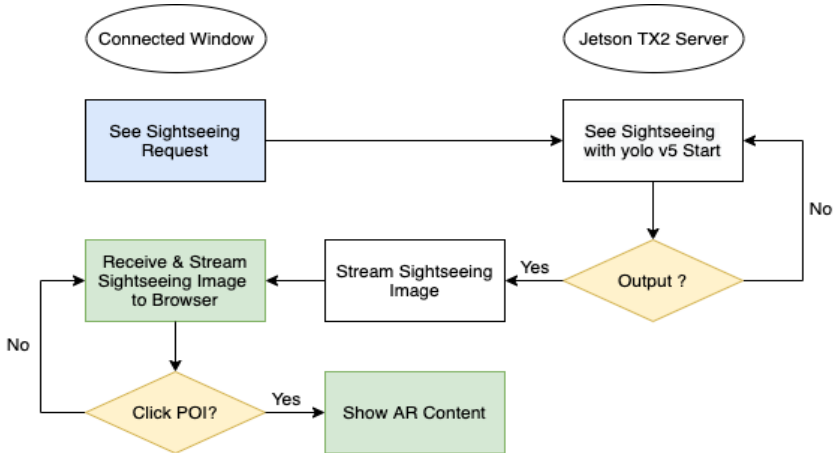


- 목적지 설정

- 교통상황 확인

- 추천 테마 소개

- 풍경 인식 및 AR 컨텐츠 제공



- 풍경인식

사용자에게 창 밖 풍경에 대한 정보를 제공하기 위해 창밖의 장면을 실시간으로 인식한다.(객체인식) 이를 통해 인식된 객체의 위치(픽셀좌표)와 객체가 무엇인지 (이름)를 사용자에게 제공한다.

- AR 컨텐츠

사용자가 커넥티드 윈도우 상에서 인식된 객체에 흥미를 갖고 클릭하면 객체정보 (좌표, 클래스)를 통해 AR컨텐츠를 제공한다.

○ Software 기능 (필요 시 알고리즘 설명 포함)

- Person Recogniton

딥러닝을 기반으로 등록된 사용자를 식별하고 사용자 맞춤 콘텐츠를 제공하기 위한 안면인식 기능을 한다.

- Sleeping Check

탑승객이 잠들었다고 판단됐을 때 수면에 방해되지 않도록 콘텐츠를 종료할 수 있게 하기 위하여 사용자의 눈이 일정 시간 이상 감겼는지 인식하는 기능을 한다.

- Search POI

탑승자가 관심을 가지는 현실 세계 또는 지도나 도면상의 특정 위치를 찾는 기능이다 또한 쉽게 목표 지점을 찾을 수 있도록 주변의 정보들도 확인할 수 있다.

- Check Road Congestion

자신의 위치에서 목적지까지의 교통 혼잡도를 통해 관광지 주변의 혼잡도 파악이 가능하고 도착 예정시간을 파악할 수 있도록 한 기능으로, 운전자와 조수석에 있는 사용자뿐 아니라 전 좌석의 사용자가 자유롭게 제어 가능하다.

- Find Travel Theme

사용자의 위치 근처에 있는 여행지를 추천하고 테마별로 정리하여 추천해 주는 기능이다.

- Sight seeing

투명 디스플레이에 AR을 띄워 사용자에게 정보를 전달하는 형태로, 차 외부의 환경과 사용자가 창문을 매개체로 실시간 상호작용할 수 있다.

커넥티드 윈도우에서 사용자가 <주변 안내> 버튼을 누르면 창밖의 이미지를 yolo v5 모델을 통해 실시간으로 객체인식 연산을 한다. 창문에 비치는 풍경에서 추천 가능한 객체가 있으면 Jetson TX2가 객체의 정보(좌표, 객체 이름)를 실시간으로 전송한다. 사용자가 이에 흥미를 갖고 POI를 터치했을 때, 해당 객체에 관련된 AR이 출력되며 사용자에게 인포테인먼트를 제공한다.

- Trunk Information

차에 탑승한 상태에서 트렁크 내부 상황에 대한 정보를 영상으로 확인할 수 있도록 하며 트렁크 내부에 있는 물체에 대한 리스트를 사용자에게 전달한다. 딥러닝 Object

Detection을 기반으로 하는 기능으로, 사용자가 간편하게 트렁크 내용물을 확인, 관리할 수 있다.

- AR

투명 디스플레이를 이용하기 때문에 증강 현실은 현실 세계에 잘 융화될 수 있다. 사용자와 가상세계 사이의 실시간 상호작용이 가능하고 높은 몰입감을 제공한다.

○ UI 사용법

- 전체 구성

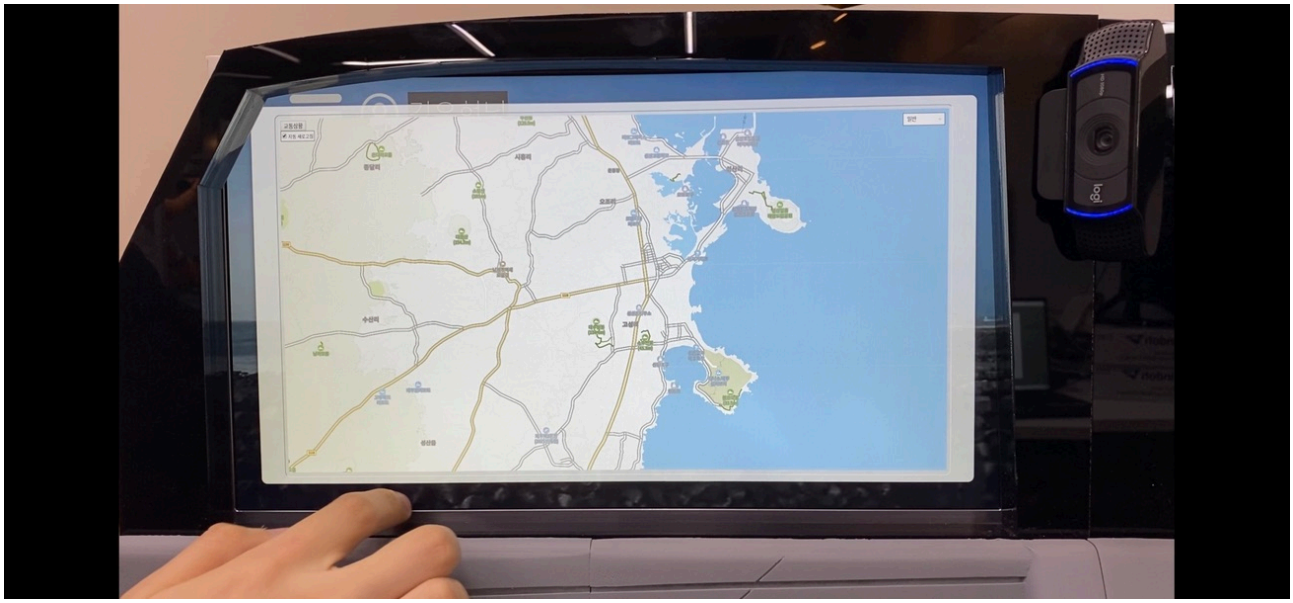
메뉴 버튼과 사용자 정보 버튼이 있다. 메뉴 버튼은 '커넥티드 윈도우'의 주요기능인 '트렁크 확인', '목적지 설정', '교통상황 확인', '추천테마 소개', '주변 안내'를 실행할 수 있는 메뉴창을 열 수 있다. 사용자 정보 버튼을 누르면 차량 내부에 설치된 카메라를 통해서 사용자를 인식한다

- 세부 구성

- 목적지 설정

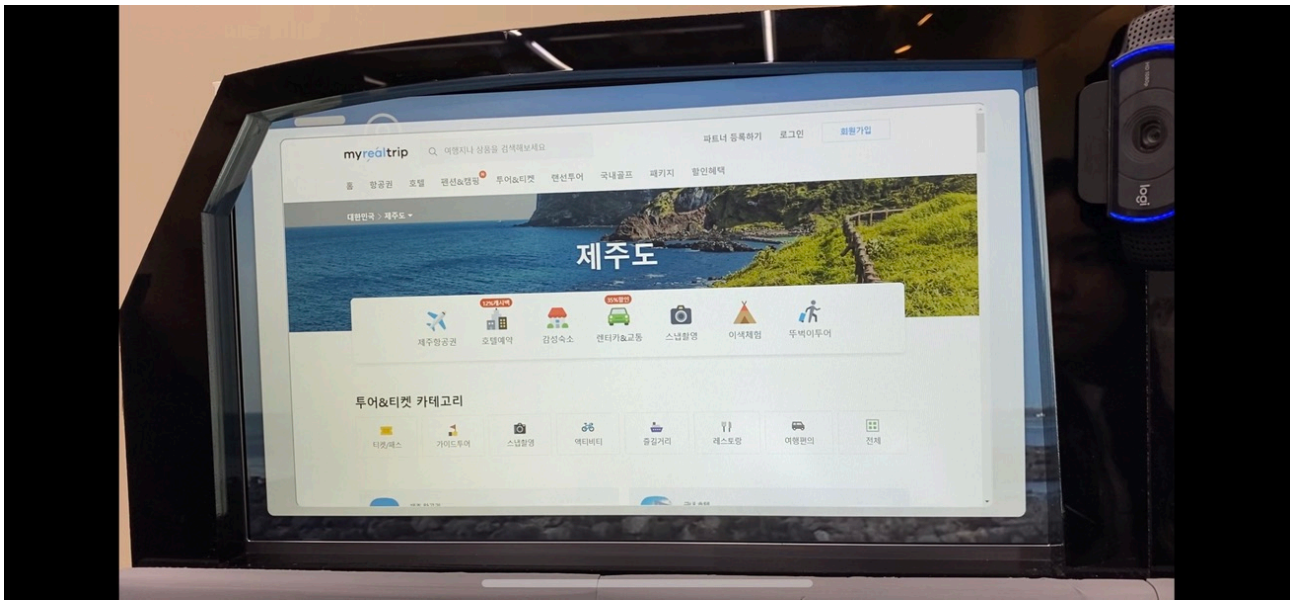
메뉴 버튼을 누른 후 목적지 설정 버튼을 누르게 되면 지도가 뜬다. 지도의 구성은 우측에는 지도, 좌측에는 연관된 장소가 나타나며 좌측상단에는 검색이 가능한 창이 있다. 검색창을 누르고 디스플레이 상에 나타난 키보드를 이용해서 원하는 목적지를 검색한다.

- 교통상황 확인



교통상황 확인 버튼을 누르면 교통상황이 표시된 지도가 나타난다. 드래그를 통해 지도를 움직이면 주변 교통상황까지 확인할 수 있다.

- 추천테마 소개

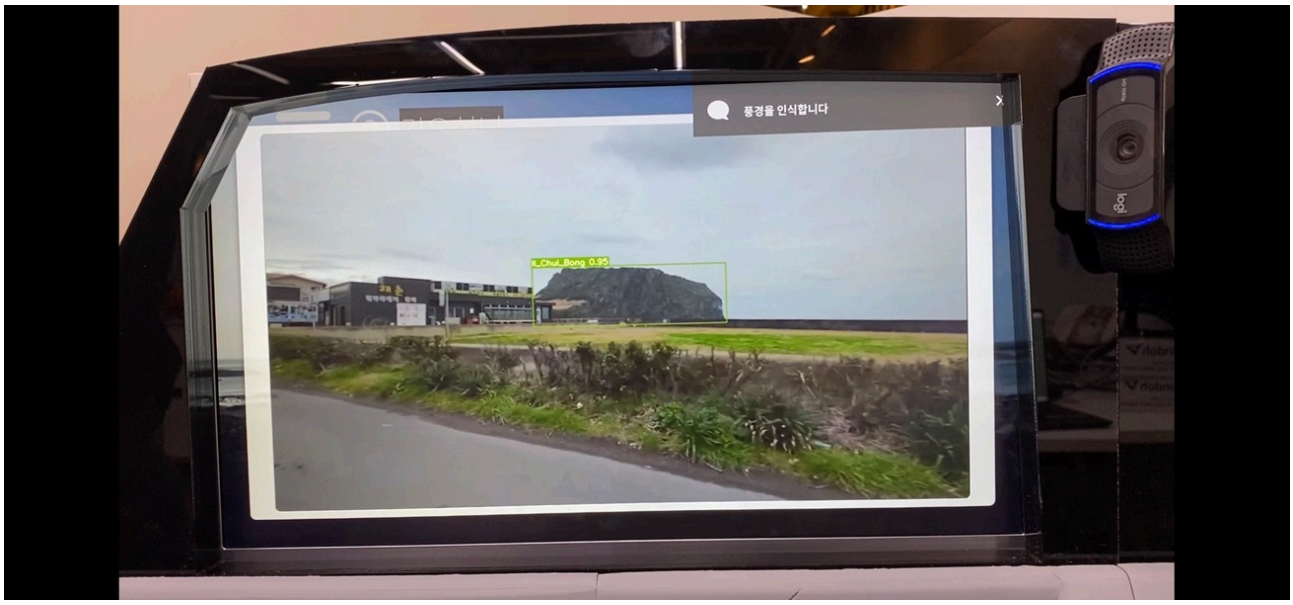


추천 테마 소개 버튼을 누른다. 새 창으로 나타난 제주도 여행 안내 사이트에서 원하는 테마에 맞는 관광지를 찾아볼 수 있다.

- 트렁크 확인

트렁크 확인 버튼을 누름과 동시에 트렁크에 장치된 카메라가 트렁크 내부를 촬영한다. 촬영한 사진을 가지고 어떤 짐이 있는지 인식을 하고 사용자에게 인식된 사진과 트렁크 내에 있는 짐 리스트를 새 창에 띄워준다.

- 주변 안내



주변 안내 버튼을 누르면 주변 인식을 시작하고 인식한 데이터를 토대로 새로운 창에서 AR 콘텐츠가 실행된다.

○ 개발환경 (언어, Tool, 사용시스템 등)

- 사용 언어, Tool

  - html, css, Javascript

  - webOS

  - webOS CLI

  - webOS LS2 API

  - Firebase

  - kakao map API

  - naver map API

  - Three.js

  - Python pytorch

    - Python imgaug

  - Python open-cv

  - Python dlib, face_recognition

  - Python Flask-socketio

  - Labelimg

  - Jetson Deep Learning Inference

  - yolo v5

- 사용 시스템

  - macOS Big Sur

  - Ubuntu 18.04

  - Windows 10

  - Nvidia Jetson TX2

  - Raspberrypi4 4GB (8GB not used becasue of bug in webOSOSE 2.7.0)

□ 개발 프로그램 설명

○ 파일 구성

- webOS

```
→  sampleApp git:(master) ✗ tree
.
├── appinfo.json
├── css
│   └── style.css
├── fonts
│   ├── NotoSansCJKkr-Light.otf
│   ├── NotoSansCJKkr-Regular.otf
│   └── NotoSansCJKkr-Thin.otf
├── icon.png
├── img
│   ├── menu.png
│   └── profile.png
├── index.html
└── js
    ├── 3d.js
    ├── flaskwebsocket.js
    ├── function.js
    └── modal.js
```

```
→  firebase2 git:(master) ✗ tree
.
├── firebase.json
└── public
    ├── 404.html
    └── index.html
```

```
→  firebase1 git:(master) ✗ tree
.
├── firebase.json
├── main.js
└── public
    ├── index.html
    ├── js
    │   └── kakaomap.js
    └── mapstyle.css
```

- Jetson TX2

```
coon@coon:~/HYMEC_EBSW_ConnectedCar/machine_learning$ tree -d
.
├── data
│   ├── images
│   └── labels
├── data_preprocessing
├── EBSW_source
│   ├── etc
│   │   ├── asdf
│   │   ├── qwer
│   │   ├── resize
│   │   ├── trunk
│   │   └── trunk_original
│   ├── media
│   │   └── movie
│   └── weights
├── inference
│   ├── images
│   └── output
├── known_image
├── models
│   ├── hub
│   └── __pycache__
├── static
├── templates
└── utils
    ├── google_app_engine
    └── __pycache__

26 directories
```

○ 함수별 기능

- Youtube player

video 태그를 이용해서 배경 영상을 재생하려고 했으나 webOS에서 video 태그가 제대로 작동하지 않아 유튜브에 업로드한 영상을 재생하기 위해 youtube API를 사용하였다.

- 지도 검색 (카카오 API)

POI를 검색하기 위해 카카오 맵 API를 사용하였다. 페이지 내 검색창에 찾고자 하는 POI를 입력하면 검색창 아래쪽에 목록으로 나오게 되고 선택하게 되면 자세한 정보를 확인할 수 있다.

- 교통혼잡 확인 (네이버 API)

교통상황 데이터를 지도에 표시하기 위해 네이버 지도 API를 사용하였다. 드래그를 통해 지도를 이동할 수 있고 도로가 혼잡하면 빨강, 혼잡하지 않으면 초록으로 표시되는 점을 활용하여 여행지 및 일반 상황에서 장소의 혼잡 정도를 파악하고 이용할 수 있도록 하였다. 같은 페이지에 동일한 API를 이용하여 표시할 때 한쪽이 변경을 시도하면 다른 한쪽이 사용을 하지 못하게 되는 문제가 발생하여 두 가지의 지도 API를 사용하였다.

- 메뉴 바

webOS 2.0 버전의 메뉴에서 영감을 받아 왼쪽 위의 버튼을 누르면 밑에서 위로 등장하는 메뉴를 구현하였다. 각각의 메뉴 페이지들은 각자의 모달을 나타나게 하고 모달이 나타나게 되면 메뉴는 최소화 상태로 줄어들게 된다.

- 모달

웹페이지를 이동하게 되면 지속적인 상태를 표시하는 등의 기능들이 불편할 것이라 판단하여 각각의 기능들을 보여주는 페이지로써 평소에는 숨어있지만 호출하게 되면 나타나는 모달을 구현하였다. 모달은 메뉴 중 하나를 누르게 되면 그에 맞는 모달이 호출되게 되고 선택한 메뉴에 알맞은 콘텐츠를 제공한다.

- AR

인포테인먼트를 가장 효과적으로 제공하기 위해 AR을 선택하였다. 블렌더를 이용해 만든 AR 캐릭터를 웹으로 불러오기 위해 webGL과 threejs을 사용하였다. 사용자가 밖의 풍경을 보다 알고 싶은 물체를 클릭하게 되면 AR 캐릭터와 TTS를 이용하여 이용자가 정보를 쉽고 재미있게 받아들일 수 있다.

- WebSocket

TX2로 대변되는 서버와 통신을 위해 웹소켓 통신을 적용하였다. 웹소켓 통신은 기존의 통신과는 달리 한번 통신으로 끊기고 다시 연결되는 것이 아니라 연결 후 지속적으로 서버와 클라이언트간의 통신이 이루어질 수 있기 때문에 사진 정보를 계속 보내기에 효과적이라 판단해 사용하였다.

커넥티드 윈도우에서는 다양한 정보를 주고받는데, 서버가 렌더링한 웹페이지를 불러올 수도 있고, 인식한 정보를 받을 수도 있다. 모든 것은 서버에 요청(Request)한 내용의 결과로 주어진다.

- Notification (LS2 API)

webOS의 LS2 API는 webOS 내의 기능들을 활용할 수 있게 만드는 API이다. 하지만 이러한 API는 webOS종속적인 앱을 만들지 않는 이상 다양한 기능들을 다 사용하기는 어렵다. 대신 주어진 알림 기능을 이용하는 것으로 API사용방법 및 확장성에 대한 가능성과 사용자에게 정보를 효과적으로 제공하였다. 알림 기능은 서버로부터 데이터를 반환하였을 때 실행되어 통신이 잘 이루어지고 있는지를 확인한다. 또한 사용자를 인식하여 알림을 띄움으로써 커넥티드 윈도우가 개인 맞춤 컨텐츠를 제공할 수 있다.

- 안면 인식: face_check()

파이썬의 dlib, face_recognition 라이브러리를 이용하여 구현하였다.

face_recognition 라이브러리는 딥러닝을 이용하여 구축된 dlib 라이브러리의 안면인식 기

능에 기반한 라이브러리이다. 사전에 등록된 사용자의 얼굴 이미지들을' load_image_file(".jpg")'모듈을 이용하여 읽어온 후 'face_encoding()'모듈을 이용해 읽어온 이미지들의 특징점(랜드마크)를 추출하여 저장한다.

여기서 하나의 이미지는 하나의 레이블(Label, 이름을 의미)을 갖는다.

사용자가 로그인 버튼을 누르면 창틀에 장착된 카메라의 프레임값을 읽어온 후 저장되어 있는 등록된 이미지의 랜드마크 좌표들과 비교를 한다. 유클리드 거리계산법을 적용하여 두 안면 랜드마크좌표의 유클리드거리가 가장 짧게 나온 이미지의 레이블값을 로그인할 사용자로 판단한다.

-수면 체크: eye_check()

사용자의 수면상태를 체크하는 함수 또한 안면인식 함수와 비슷한 방식으로 구현되었다.

안면인식 프로그램이 두 이미지의 안면 랜드마크 좌표 간의 관계를 토대로 연산을 한다면 수면체크 함수는 사용자의 눈꺼풀사이 거리를 토대로 연산을 진행한다. 커넥티드 윈도우가 재생되고 있을 때 창틀에 장착된 카메라는 사용자의 얼굴을 지속적으로 읽어온 후 face_recognition 라이브러리를 이용하여 사용자의 얼굴에서 눈가, 눈동자, 눈썹 등의 랜드마크(특징점)을 추출한다. 이를 통해 두 눈꺼풀사이의 거리(유클리드 거리)를 구할 수 있고 조건문을 통해 두 눈이 모두 일정 비율이상 감겼을 때 웹소켓 메세지를 전송하여 컨텐츠를 종료할 수 있게 구성하였다.

-객체 인식: YOLOv5

YOLOv5모델은 파이썬의 pytorch 프레임워크를 기반으로한 Object Detection모델이다. YOLOv5모델은 빠른 연산속도와 정확도를 가진다. 학습데이터에 알맞는 Anchor Box를 자체적으로 계산해주므로 학습데이터 전치리 과정이 비교적 간단하다. 머신러닝의 특성상 다량, 양질의 데이터가 모델의 성능을 높이므로 직접 데이터를 수집한 후 라벨링하였

고 YOLOv5모델에 적합한 데이터 증강기법을 적용하였으며 창밖 풍경(Sightseeing)과 트렁크(Trunk) 영상을 인식하기위해 모델을 두 가지로 나누어서 학습시켰다.

창밖의 풍경을 인식하기 위한 모델은 제주도의 관광 요소중 4개를 선정하였으며 제주도에 답사를 가서 직접 촬영한 영상과 구글 크롤링 데이터, 구글 로드뷰 캡쳐 데이터를 이용하여 학습데이터를 수집한 후 라벨링했다. 트렁크를 확인하기 위한 모델은 여행을 떠날 때 챙기는 물건들 중에서 5가지를 선정하여 데이터를 직접 수집한 후 라벨링하였다.

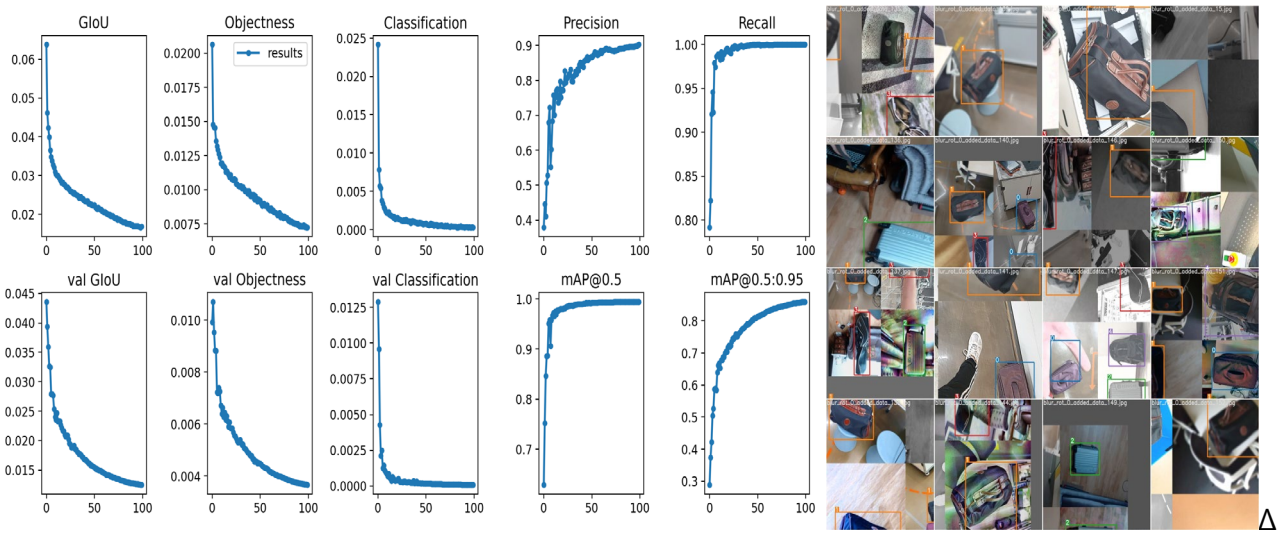
파이썬 imgaug 라이브러리를 사용하여 데이터 증강기법을 적용했으며 이를 통해 학습데이터량을 20배가량 증가시켰다. 데이터 증강 기법으로는 기존의 YOLOv4논문을 참고하여 YOLO모델 성능향상에 도움을 주는 "Rotation", "Blur", "Gray", "Contrast", "Embossing" 총 5가지를 사용하였다.

학습은 "Sightseeing model" : 12654장, "Trunk model" : 34220장의 학습데이터를

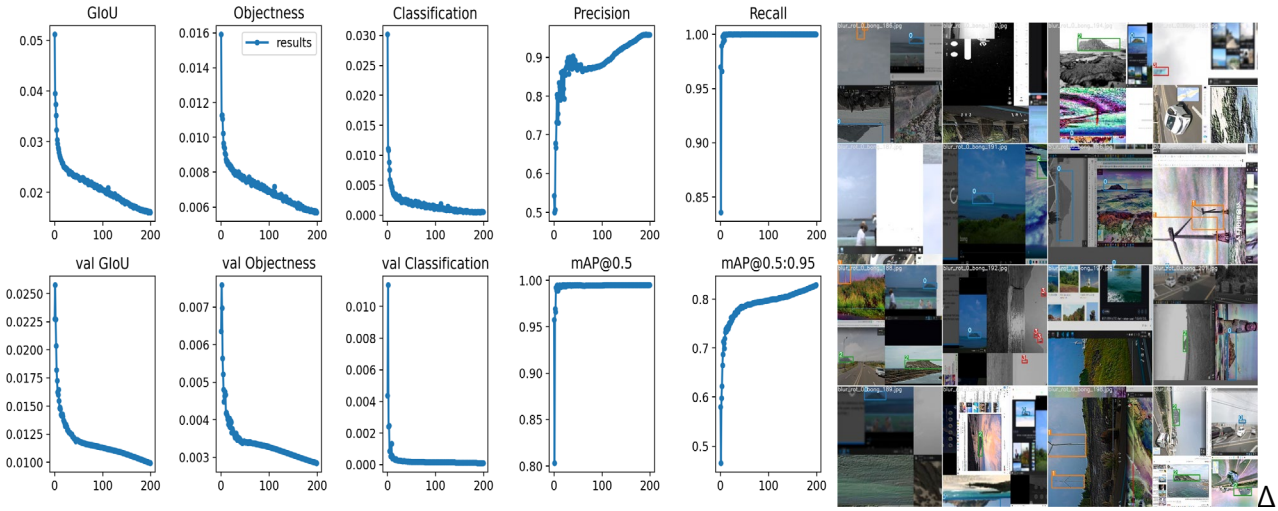
32 batch size, 200 epoch의 하이퍼파라미터로 학습시켰으며

창밖 풍경 인식모델은 mAP@(0.5:0.95) = 0.9023, ::(mAP : mean Average Precision)

트렁크 확인 모델은 mAP@(0.5:0.95) = 0.8863 의 정확도까지 각각 학습시켰다.



Sightseeing model results & train batch

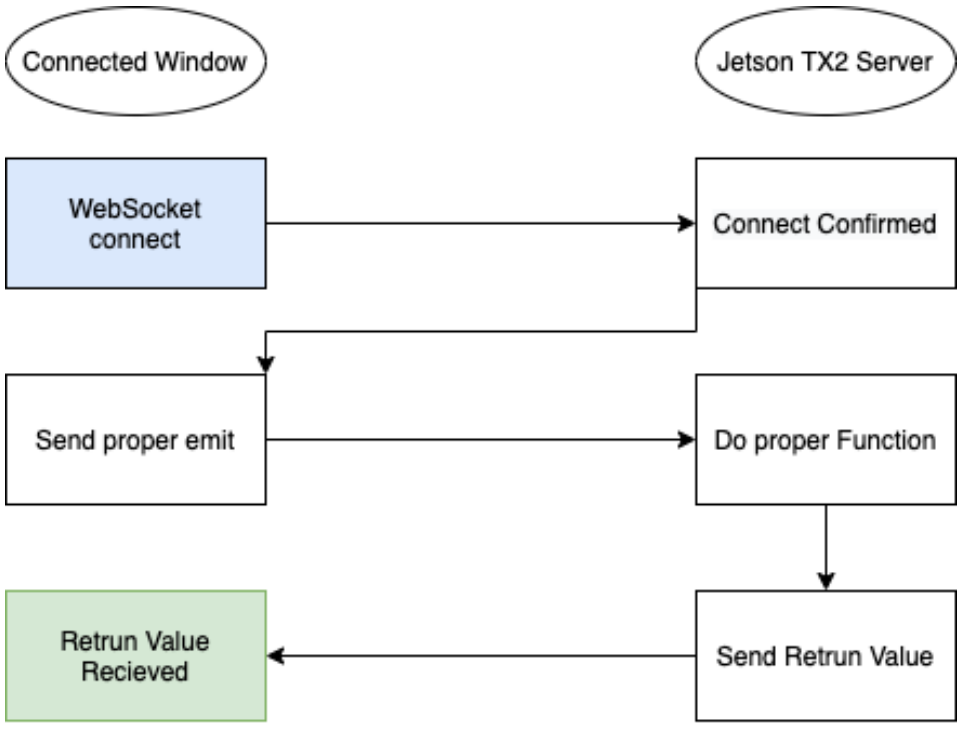


Trunk model results & train batch

학습시킨 두 모델은 Jetson TX2에서 최대 15fps의 연산속도로 연산 가능하다.

○ 주요 함수의 흐름도

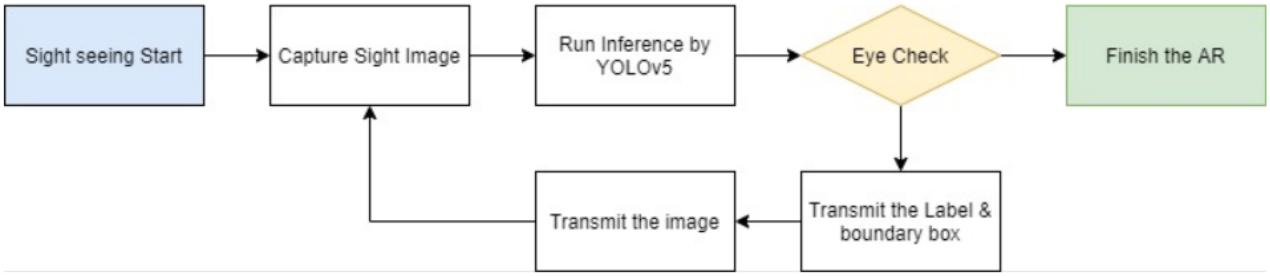
- WebSocket



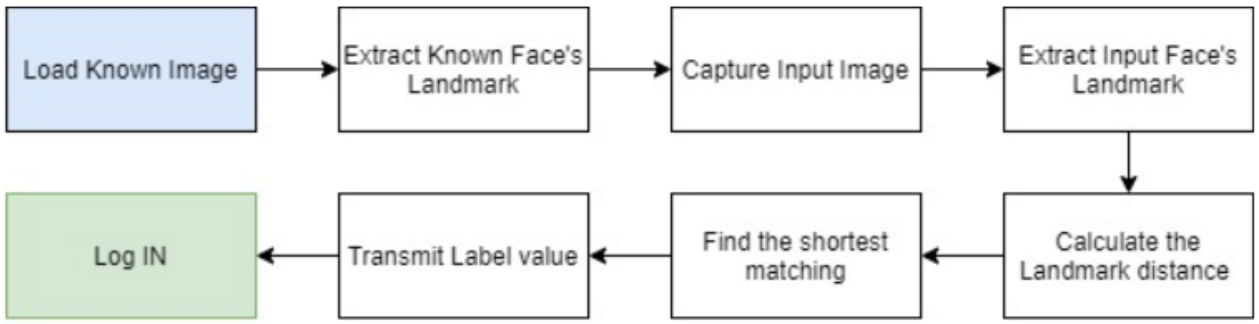
webOS에서 서버로부터 적합한 결과를 제공받기 위해 Handshake 과정을 통해 연결을 확인 후 Flask의 메세지 전송기능인 emit함수를 이용하여 헤더와 인자들을 전송한다.

전송받은 인자들을 이용해 TX2가 작업을 선정하고 실행 후 결과값을 webOS로 반환하여 커넥티드 윈도우에 표현한다.

- Yolo v5 (이미지 인식)



- 얼굴인식



○ 기술적 차별성

- 웹에서 표현되는 머신러닝

머신러닝 서비스를 웹에서 구현함으로써 사람들이 머신러닝을 접할 수 있는 접근성을 높였으며 머신러닝을 활용하여 다양한 기능을 체험할 수 있는 기회 또한 확장했다.

- 발전 가능한 머신러닝

도로상의 객체를 인식하여 활용할 수 있는 방안은 무궁무진하다. 도로상의 차량들과 로드뷰 데이터 등을 활용하여 모델을 학습시킨다면 다양한 객체들(간판, 랜드마크, 브랜드 로고 등등)을 이용하여 탑승자에게 더욱 흥미로운 인포테인먼트를 제공할 수 있다.

- webOS를 뛰어넘은 webOS

webOS는 현재 TV 및 가전에서 활용되고 더욱 더 사용범위가 넓어지고 있다. 이러한 상황에서 머신러닝을 더하여 다양한 서비스를 제공할 수 있다는 것을 입증함으로써 활용 범위를 기존의 영역보다 넓히게 되었다.

□ 개발 중 장애요인과 해결방안

○ 하드웨어 제작 상의 장애요인과 해결 방안

- 프린터 출력 최대 크기보다 출력물이 더 커서 8파트로 나눠서 출력을 진행했다. 하지만 3D프린터 특성상 품질이 항상 균일하지 않기 때문에 파트가 조금씩 왜곡이 생긴다. 그래서 변형이 보다 적은 abs 필라멘트를 사용하여 사포질 등의 후가공으로 마무리하였다.

- 3D 프린터는 복잡한 형상을 구상할 수 있지만 그만큼 기회비용이 크다. 이 때문에 아크릴과 나무판자를 추가적으로 활용하여 기회비용을 줄이고자 했다.

- 디자인적으로 제품을 '차 내부'라는 것을 전달하는 것이 쉽지 않았다. 실제 차량 내부는 벽이 곡선으로 안쪽으로 휘어 있으며 곡선이 많다. 이를 위해 3D프린팅만을 이용해서 내부를 표현해보고 싶었지만 금전적, 시간적 비용이 커서 뼈대(아크릴)에 차량 내부 디자인과 비슷하게 출력한 3D프린터를 최소한으로 사용해서 표현했다.

○ webOS, 라즈베리파이4 및 젯슨 보드의 성능한계

- 더욱 더 다양한 기능을 구현해보려 하였으나 임베디드 보드 및 OS의 한계로 인하여 더 다양한 것을 넣지 못하였다. 이것은 임베디드 개발자들이 더욱 더 고민해야 할 문제이고 앞으로 임베디드 보드에서 동작할 수 있게 포팅 또는 최적화를 시켜 나가야 할 것이다. 우리는 기능을 제한시키거나 한 기능을 동작 시에는 다른 기능이 동작하지 않게 만들어 최대한의 컴퓨팅 파워를 구동하는것으로 해결하였다.

○ 임베디드보드의 인공지능 알고리즘 연산 성능 한계

- 개발 초기단계에는 머신러닝 기능을 구현하기 위해 Nvidia Jetson Nano 보드를 이용하였지만 프로그램을 개발한 후 모델을 학습시켜 실제 데이터에 적용시켜보니 연산속도가 느리다는 한계점이 있었다(YOLOv5 기준 4~5fps) 따라서 GPU성능이 더욱 뛰어난 Jetson TX2보드로 교체하였고 최대 15fps의 연산속도까지 향상시킬 수 있었다. 하지만 객체인식 연산과 웹서버 통신을 병행할 시에는 이보다 연산속도가 느려지므로 Jetson TX2보드도 완벽한 해결책이 되지 못했다. 우리의 아이디어를 완벽히 구현하기 위해서는 창밖에서 변화하는 풍경을 실시간으로 연산하여 이를 사용자가 괴리감없이 느껴야 하는데 이를 위해서는 20fps이상의 연산속도가 필요하므로 더 좋은 컴퓨팅성능을 갖는 임베디드보드가 요구된다.

□ 개발결과물의 차별성

○ 컨텐츠 다양성

'커넥티드 윈도우'의 주요 기능인 '주변 안내' 기능은 차량의 현재 위치 주변의 정보 (유적지, 맛집 등)를 실시간으로 수집하여 그 객체에 적합한 콘텐츠를 제공한다. 성산 일출봉이 창밖에 보이면 주변 안내를 통해 제주도 캐릭터 AR의 설명을 들을 수 있다. 핸드폰만 있으면 제공받을 수 있는 단순한 콘텐츠가 아니고 주변의 환경이 실시간으로 콘텐츠가 되어 사용자가 인포테인먼트를 즐길 수 있다.

○ 안전성 및 사용성

최근 LG에서 'Conniro'를 커넥티드 카 솔루션으로 내놓았다. 'Conniro'의 가장 큰 특징은 차 앞 유리에 증강현실을 구현하여 운전자가 길을 찾고 안전하게 운전하기에 수월할 수 있는 기능을 가지고 있는 것이다. 하지만 다른 차, 보행자 등 수많은 요소에 신경 써야 하는 운전자가 바라보는 창에 AR을 구현한다면 안전성에 있어서 좋지 않은 효과를 가져올 것이다. 또한, 앞 창문에 AR을 구현하므로 이용성에 있어서도 직관적이지 않을 수 있어 사용자가 불편함을 호소할 수 있다. 하지만 이 기능을 '커넥티드 윈도우'와 같이 측면 창문에 가져온다면 운전자의 시야에 방해되는 요소를 줄일 수 있어 'Conniro'의 솔루션보다 더 나은 안전성을 가진다. 터치로 제어하는 방식을 사용할 수 없는 정면 창에 비해 손 닿기 용이한 위치에 있는 측면 창문이 사용자에게 조작의 편의성을 선사한다.

○ 심미성

요즘에 나오는 고급 차량 옵션으로 뒷좌석 디스플레이가 설치되어 있다. 아직 이러한 디스플레이들은 심미적으로 차량 시트와 잘 어우러진다고 보긴 어렵다. '커넥티드 윈도우'는 앞 좌석 의자 뒤에 장착되어 있던 디스플레이가 차량 측면 창문 자체로 대체되기 때문에 심미성을 해치지 않는다. 태블릿 pc가 좌석 뒤에 붙어있는 수준과는 달리, 사용자는 주변 광경과 정보, 콘텐츠들에 둘러싸여 인포테인먼트를 즐길 수 있다.

□ 개발 일정

No	내용	2020年															
		6月				7月				8月				9月			
	프로토타입 모델링					■	■	■	■	■	■						
	최종 하드웨어 모델링										■	■	■	■	■		
	하드웨어 제작/가공														■	■	■
	AR 컨텐츠 개발												■	■	■	■	
	웹 앱 개발			■	■	■	■	■	■	■	■	■	■		■	■	■
	YOLO 사용법 숙지				■	■											
	객체인식 데이터 수집										■	■	■	■			
	YOLOv5 모델 학습												■	■	■	■	
	통신 구축															■	■

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	김유현	webOS, 웹 앱, 기획
2	팀원	정석훈	머신러닝 : 얼굴인식, 객체인식
3	팀원	권소은	하드웨어 설계, 가공
4	팀원	이동규	하드웨어 설계, 가공
5	팀원	양지윤	AR 컨텐츠 제작 및 시나리오 작성